PROJECT NAME: Case Report System
INSTRUCTORS: Jacky Tang
STUDENT: Jonny Ieong (P-01-0582-6), Cleskey Cheang ().
HAND-IN DATE: 2006-01-24

- Abstraction 5
- Introduction 5
- Objectives 5
- The process of reporting a case 6
- Data Communication 7
  - Explanation of steps on the above diagram 7
  - How does the server or client trust the remote authentication credentials 9
- Background Information 9
  - Data Encryption 9
  - Digital signature 9
  - SSL 10
  - Symmetric Key Pair Encryption 10
  - Smart Card 10
    - Contact Smart Card: 11
    - Contactless Smart Cards: 11
    - Memory cards and microprocessor cards 11
  - Protocol T = 0 11
  - Protocol T = 1 11
- Smart Card Specifications: 12
- Smart Card Reader Specifications: 13
- System Requirements: 13
- Installation Guides: 14
  - Server 14
  - Client 14
- User Manual 15
  - Server 15
  - Client 18
    - Write User information 19
    - Change PIN code 19
    - Extract KeyStore 20
    - Communication with Server and report the cases 20
    - Report cases 21
- Class structure and illustration 22
  - Server 22
    - access_database.java 22
      - register 22
      - insert_analysis 22

retrieve_puk 23
register 23
insert_complaint 23
follow_complaint 23
view_complaint 23
view_status 23
view_analysis 23
Analysis_packet.java 24
analysis_string 24
analysis_cert 26
analysis_byte 26
converter.java 27
convert_to_byte_array 27
convert_to_publickey 27
establish_ssl.java 28
getServerSocket 28
get_certificate 28
get_privatekey 29
get_publickey 29
Process_Connection.java 29
server.java 30
Signature_verifier.java 31
verify_sig 31
Client 32
CRException.java 32
SmartCardManager.java 33
changePIN 35
initCardStructure 35
initUserDataFile 35
isCardInserted 35
readXXX 36
writeXXX 36
CntThread.java 36
run 36
getConnected 37
setConnected 37
UserDataDefinition.java 37
WriteCardManager.java 37

writeCardInfo....38
isBlankCard....38
DeploManager.java....38
checkPin....38
connectToServer....38
readCardInfo....39
sendComplaint....39
viewComplaint....39
KeyStoreManager.java....39
getKeyStoreInByte....40
loadKeyStoreFromByte....40
PopAction.java....40
TextAreaPopup.java....41
DesktopPopup.java....42
BuildTableModel.java....42
InterfaceManager.java....42
frameActivate....43
closeFrame....43
closeAllFrames....43
tileVertical....43
tileHorizontal....43

✹Abstraction

In Macau, residents always ask for improvement on any social activities. They are complaining the police govern department doesn't provide enough ways for them to report cases and it is a true that we found there are not developed any services of reporting cases to police and also Smart ID Card is commonly use by Macau resident. So, we will provide residents a secure convenience environment to report cases to police that is the main objective in this project.

✹Introduction

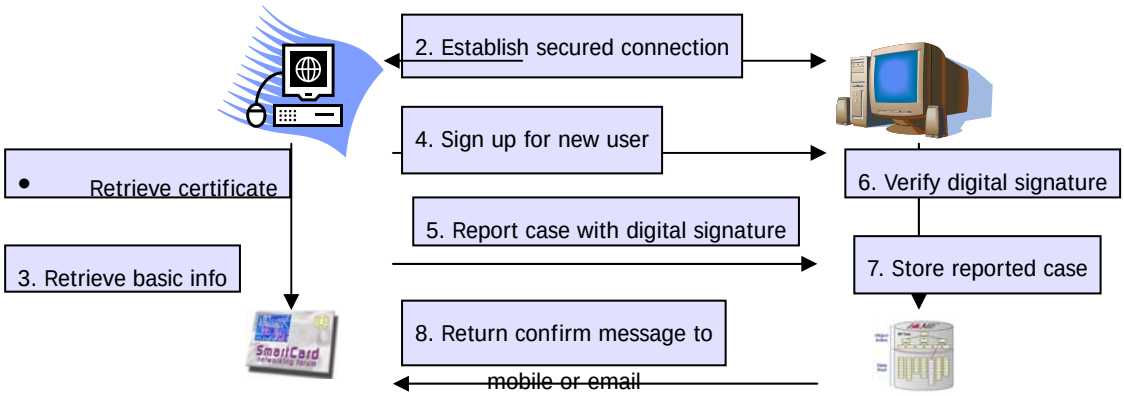
Computers are very common in the all over the world; most corporations already transferred or intend to put their services into the Internet, e.g. E-commerce, apply for license or gambling on the Internet. Internet services are quickly been developing because it brings convenience, secure, easy to use and reliable to user.

We assemble security and Smart Card technology in order to provide a secure, convenience gateway and also provide an reliable and easy to use environment to citizens reported their cases to police. That is our objectives in the project.

✹Objectives

※Citizens can make a valid case reporting without face-to-face situation via a secured internet service.
※The police can still authorize the identity of reporter via the digital signature verification.
※Suggest police develop this service and let the citizens have one more way to report cases on corruption.
※Promoting portable Smart Card as the media in the application.
※Easy-to-use, reliable, convenience and cost effective.

✹The process of reporting a case


2. Establish secured connection
4. Sign up for new user
6. Verify digital signature
Retrieve certificate
5. Report case with digital signature
3. Retrieve basic info
7. Store reported case
8. Return confirm message to
mobile or email

The reporting progress is made through a secured communication media.
Its characteristics are listed as following:

※Encryption

Protects data from unauthorized use by converting it to an apparently meaningless form before transmission.

※Source authentication

Verifying the data sender's identity. The server presents the client with a set of credentials in the form of a certificate, the client does so.

※Data integrity

Data has not been modified in transit.

※Database in Server

The data received from the sender is encrypted by secret key before stored into the database.

Access password is needed.

Once the database is really opened by somebody. Records show in ciphertext.

✸Data Communication

Explanation of steps on the above diagram

1)Client runs the client application and send a connection request to the server.

2, 3) Server responses to the client by send the client its certificate.

4) At the same time, the server also requires for the client certificate for mutual authorization

5) According to the SSL mechanism, there is a key exchanged for the data encryption during the data transition.

7) The client sends its certificate to the server for authorization.

8) Key exchanged for the data encryption.

9) The server versifies the certificate by checking whether its issuer is a authoritative CA.

10-12) Both Server and Client inform each of others their supported cipher specification.

14) The SSL Handshaking is finished. The case reporting progress can be secured by the Secured Socket Layer mechanism.

Please note that:

Encryption protects data from unauthorized use by converting it to an apparently meaningless form before transmission. The data is encrypted by one side (the client or the server), transmitted, decrypted by the other side, then processed.

Source authentication is a method of verifying the data sender's identity. The first time

a browser or other client attempts to communicate with a Web server over a secure connection, the server presents the client with a set of credentials in the form of a certificate.

Certificates are issued and validated by trusted authorities known as certification authorities (CAs). A certificate represents the public-key identity of a person. It is a signed document that says: I certify that the public key in this document belongs to the entity named in this document. Signed (certificate authority). Well-known CAs include Verisign, Entrust, and Thawte. Note that the certificates used with SSL/TLS today are X.509 certificates.

Data integrity refers to means of ensuring that data has not been modified in transit.

Authentication and Non-Repudiation with Digital Signatures

A digital signature provides a mechanism for authentication and non-repudiation, as well as strong data integrity. Let's look at a couple of scenarios:

A user must share a sensitive document with a given recipient. In this case the recipient needs proof the document in fact came from the expected user. Before sharing the document, the sender signs the document using his private key. The sender had previously shared his public key with the recipient, who uses it to verify sender's signature.

A user must prove its identity before it is allowed to consume some services on the Internet. Typically usernames and passwords have been used for this. Because a digital signature contains signed (protected) information such as distinguished names that uniquely identifies a user, digital signatures provide a very strong approach to authentication.

A transaction, monetary or not, is initiated by a user from a J2ME application. This transaction requires support for non-repudiation, which means that once the transaction is approved by the user and processed on the other end, it cannot be repudiated or rejected on the basis that "it never happened or is not valid."

Non-repudiation can be guaranteed by signing the transaction using a digital signature. Non-repudiation-type signatures are special because they are indented to legally bind a user to a transaction.

When transmitting sensitive information, detecting data tampering is important. Because a digital signature is based on signed message digests, the result is authenticated, strong (cryptographic) data integrity and data tampering detection.

How does the server or client trust the remote authentication credentials

A trust manager is responsible for determining if the remote authentication credentials should be trusted. The following rules are used:

If a truststore is specified by the javax.net.ssl.trustStore system property, then the trust manager will use the file provided to check for credentials. If, however, the system property exists but the file specified doesn't exist, then no truststore is utilized and a CertificateException will be thrown.

If the javax.net.ssl.trustStore system property is not defined, then a default trust store is searched for:

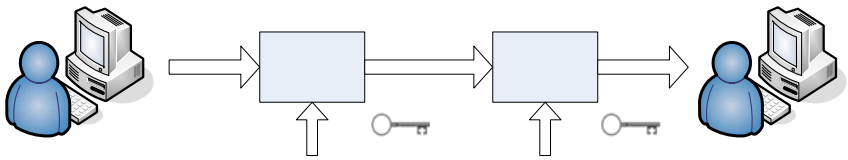
If a trust store named jssecacerts exists in the lib/security subdirectory of your java.home directory, it will be used.

If jssecacerts doesn't exist, but cacerts does (which is shipped with the J2SDK with a limited number of trusted root certificates), it will be used.

✸Background Information

Data Encryption

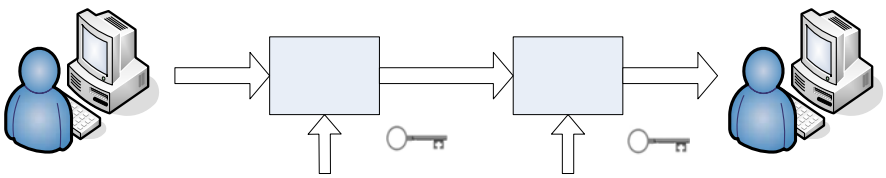
Use public key to encrypt and private key to decrypt critical information, Usage: keep data away from unauthorized user.



The above figure shows an illustration of the usage of Data Encryption in asymmetric cryptography. Data that is encrypted with the public key can be decrypted with the private key.

Digital signature

Use private key to encryption and public key to decryption. Usage: Identify and verify someone.



The above figure shows an illustration of the usage of Identification and Verification in

asymmetric cryptography. Data that is encrypted with the private key can be decrypted with the public key.

SSL

SSL, short for Secure Sockets Layer , A protocol for transmitting data securely over the World Wide Web. Whereas SSL creates a secure connection between a client and a Server, over which any amount of data can be sent securely, S-HTTP is designed to transmit individual messages securely.

The Secure Sockets Layer protocol is a protocol layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP).

One common use of SSL is to secure Web HTTP communication between a browser and a webserver. This case does not preclude the use of non-secured HTTP. The secure version is mainly plain HTTP over SSL (named HTTPS), but with one major difference: it uses the URL scheme https rather than http and a different server port (by default 443).

Symmetric Key Pair Encryption

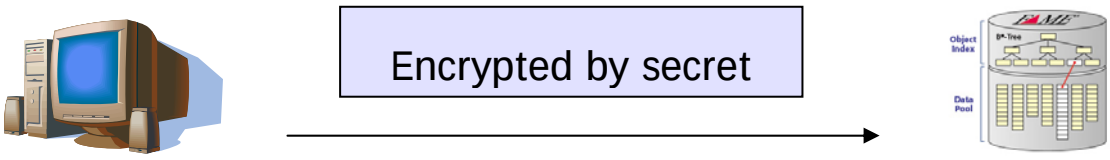
Some critical database records are secured by using the symmetric key pair encryption.

The critical data received from the sender is encrypted by secret key before stored into the database.

Database access password is needed.

Once the database is really opened by somebody. Records show in cipher text.

Only critical data records will be encrypted instead of encrypting the whole database since encryption / decryption increase processor overheads.



Smart Card

What is Smart Card?

Identical to the size of a credit card, a smart card stores and processes information through the electronic circuits embedded in silicon in the plastic substrate of its body.

There are three types of Smart Card.

Contact Smart Card:

Contact smart cards must be inserted into a smart card reader. They have a small gold plate about quarter inch diameter on the front, instead of the magnetic strip on the back like a credit card. When the card is inserted into a smart card reader, it makes contact with electrical connectors that transfer data to and from the chip.

Contactless Smart Cards:

Contactless Smart Card is passed near an antenna to carry out information exchanges.

Memory cards and microprocessor cards

Smart cards come in two varieties: memory cards and microprocessor cards. Memory cards simply store data and can be viewed as a small floppy disk with optional security. A microprocessor card, on the other hand, can add, delete and manipulate information in its memory on the card. Similar to a miniature computer, a microprocessor card has an input/output port operating system and hard disk with built-in security features.

Protocol T = 0

The T=0 protocol is a byte-oriented protocol where a character is transmitted across the channel between the reader and the card. In addition, error handling is performed on each byte by looking at the parity bit. If the actual parity bit does not correspond to the parity of the transmitted data, then an error must have occurred. In the T=0 protocol, the receiving side signals that it requires the byte to be retransmitted in the case of detecting a parity error. When the transmitting side detects this, it resends the byte that was previously received in error.

Protocol T = 1

The T=1 protocol is a block-oriented protocol in which a well-defined collection of data or a block is moved as a unit between the reader and the card. This block may incorporate an application-specific APDU providing an example of the distinct layering between the link and application protocol layers. However, the error detection and correction is more complex than that used in the T=0 protocol.

✹Smart Card Specifications:

Contact Smart Card

ACOS1 revision 3.0

8 Kbytes of EEPROM memory for application data

Compliance with ISO 7816-3, T=0 protocol

Five secret codes + Issuer Code

PIN, changeable by card holder

Session key based on random numbers

Linear files with fixed record length; record length can be different for different files

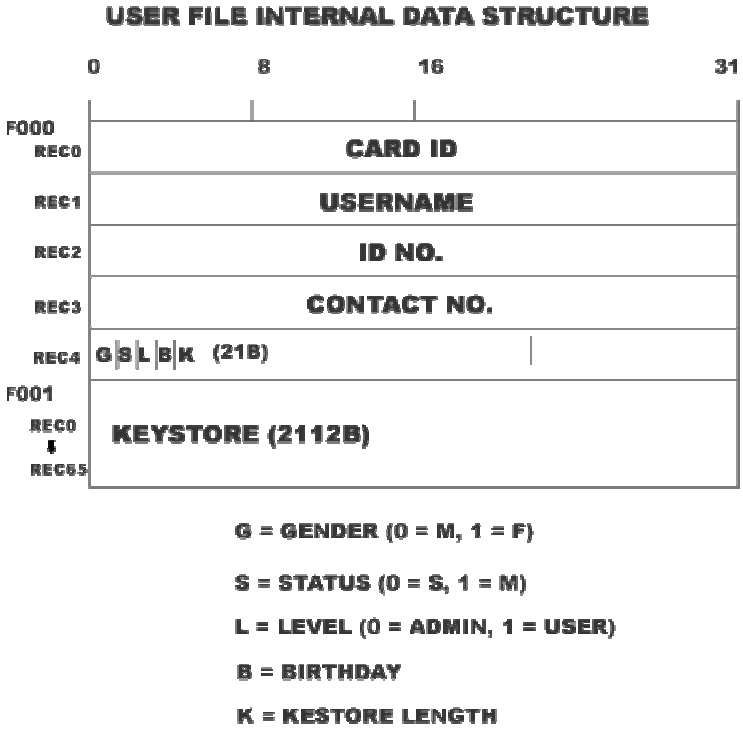
Account data structure for highly secure payment applications as an optional function

Encrypted PIN and Issuer Code when transmit or receive from/to card accepting device

According to the guideline provided by ACOS, we follow the below structure to read / write our Smart Card. Please refer document\img\InternalFileStructure.jpg

		File Security Attributes			Record Organization
Memory Area	Internal File ID	Manufacturing Stage	Personalization Stage	User Stage	
MCU-ID File	FF 00_H	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	2 x 8 bytes
Manufacturer File	FF 01_H	R: FREE W: IC	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	2 x 8 bytes
Personalization File	FF 02_H	R: FREE W: IC	R: FREE W: IC	R: FREE W: NO ACCESS	3 x 4 bytes
Security File	FF 03_H	R: IC W: IC	R: IC W: IC	R: NO ACCESS W: IC	12 x 8 bytes
User File Management File	FF 04_H	R: FREE W: IC	R: FREE W: IC	R: FREE W: IC	N_OF_FILE x 6 bytes
Account File	FF 05_H	R: FREE W: IC	R: FREE W: IC	R: IC W: IC	8 x 4 bytes
Account Security File	FF 06_H	R: FREE W: IC	R: FREE W: IC	R: NO ACCESS W: IC	4 x 8 bytes
User File Data Area	file IDs: xx yy_H xx ≠ FF_H	according to the file definitions			

And the following is our card format according to ACOS.



Remark: We store the KeyStore into memory because of security reason and convenience.

✹Smart Card Reader Specifications:

ACS30 Card Reader
USB type connection
Fixed speed 9600bps
Provide C language API to be called, because of this, we need JNI (Java Native Interface) as an intermediate to call up C program.
ACS30 proprietary driver

✹System Requirements:

Windows 9x, 2000, XP
Java 1.4 version
Tomcat 5 Web Server
ACR30 Proprietary Driver
Advanced Card Systems Ltd. ACR30 Smart Card Reader
ACOS1 revision 3.0 Smart Card

✹Installation Guides:

For the experiment purpose, this project assume that you install all component in one computer, and you have to use the localhost as the internet address. For java JDK and JRE the version must be 1.4 or above

Server

For tomcat the version must be 5 or above

1. You must install the MySQL Server. (You can get the latest version from www.mysql.com. We recommend you to download MySQL Server 5.0)
   Copy all files/folders in folder schema into directory: C:\Program Files\MySQL\MySQL Server 5.0\data\. It is the directory where MySQL database scheme placed.
2. Copy the jar file named mysql-connector-java-3.1.12-bin.jar into .\lib\ext\ under your java home directory. (e.g. C:\JBuilderX\jdk1.4\jre\lib\ext).
3. Install JBuilder10 or higher version.
4. Simply open the file ccac.jpx in the server_install folder and the JBuilder will open it. Click the run button (green arrow) on the JBuilder tool bar and then click “server” from the pop-up list. The server will run.

Client

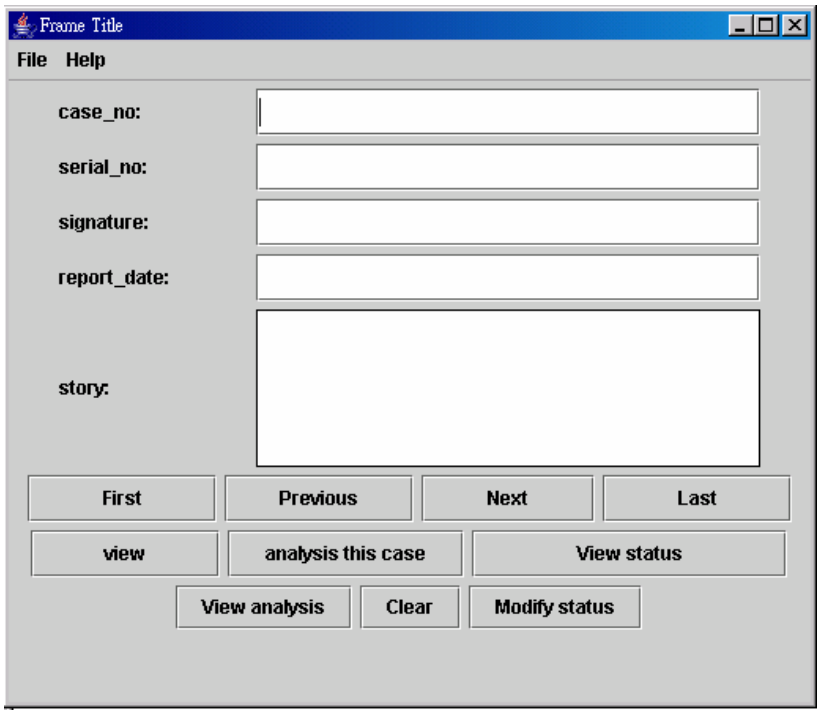
(Note: if you had installed before you can skip this)

1. Open the folder “\Third party\ACS driver\” from the CD and to be installed
   Run the setup program
   During the installation choose the smart card reader model “ACR30U”
2. Create a folder named you want (our example name Project).
3. Copy the folder “Third party” and create a folder (our example name Self) in Project.
4. Create a folder named you want again(our example name J) in Self folder
5. Copy the CEO.jar and img folder to that folder.
   Double click CEO.jar and run our program.

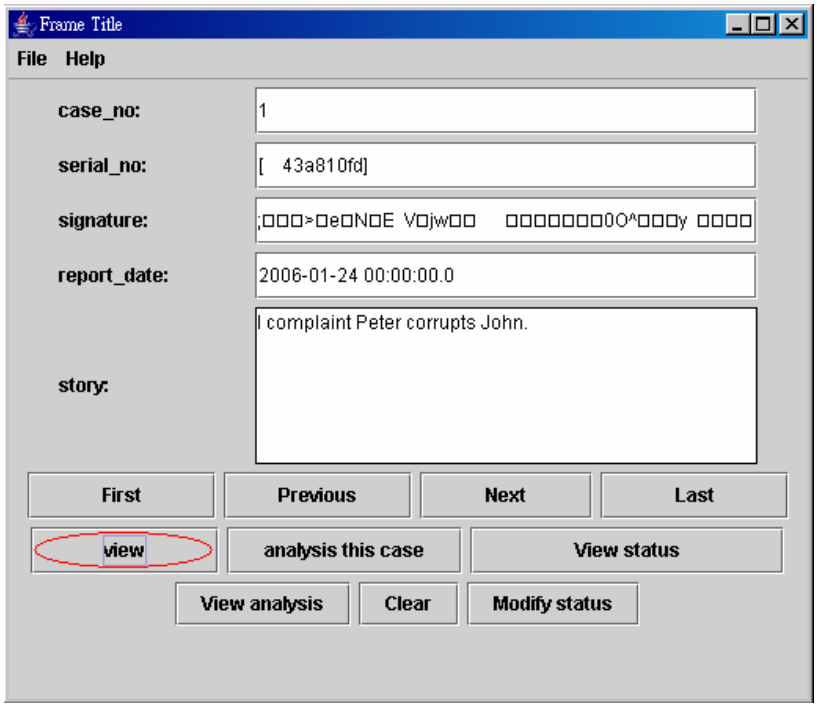
✽User Manual

Server

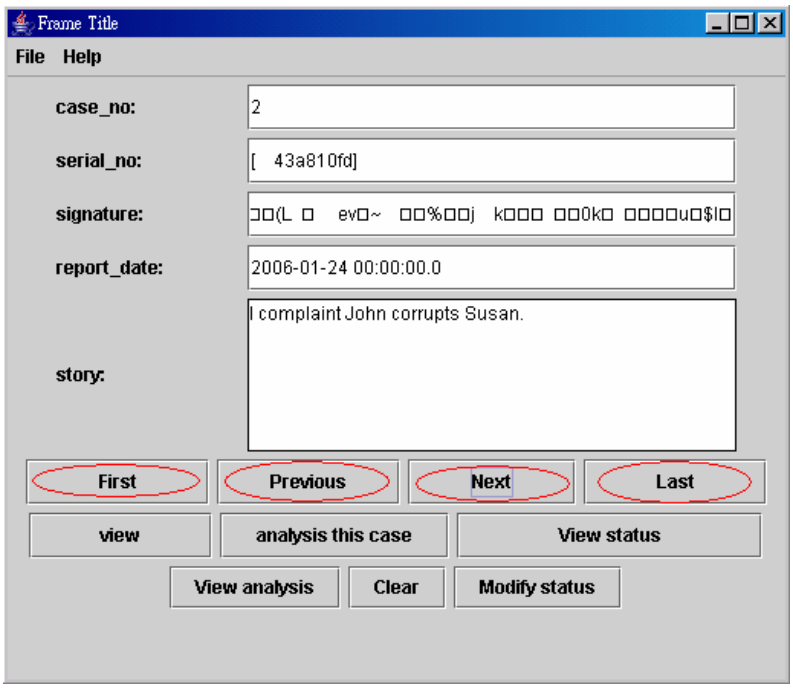
This is the main control panel of the backend client for C.C.A.C. staff.



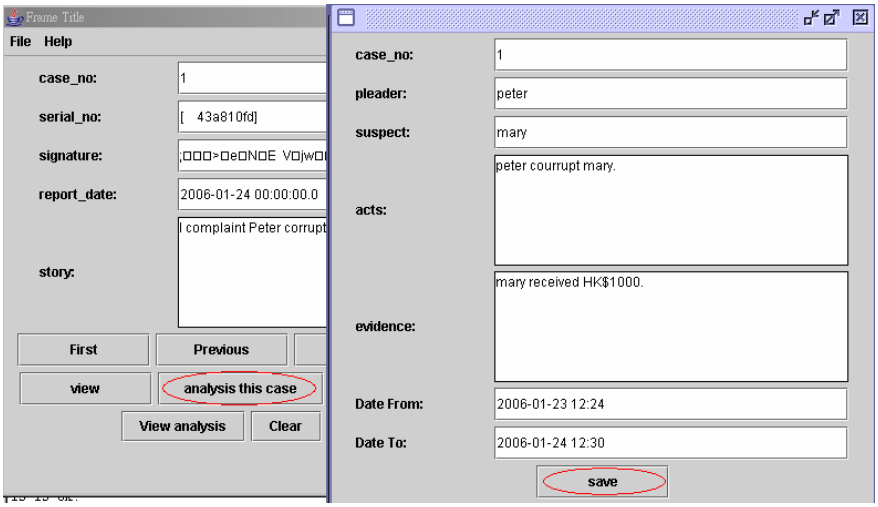
click the view button to show the reported cases from residents.



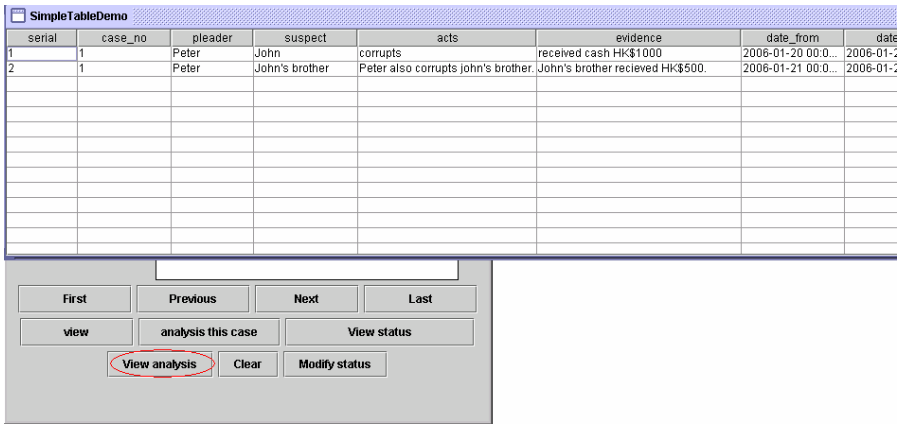
click button previous, next, first or last to navigate all reported cases.



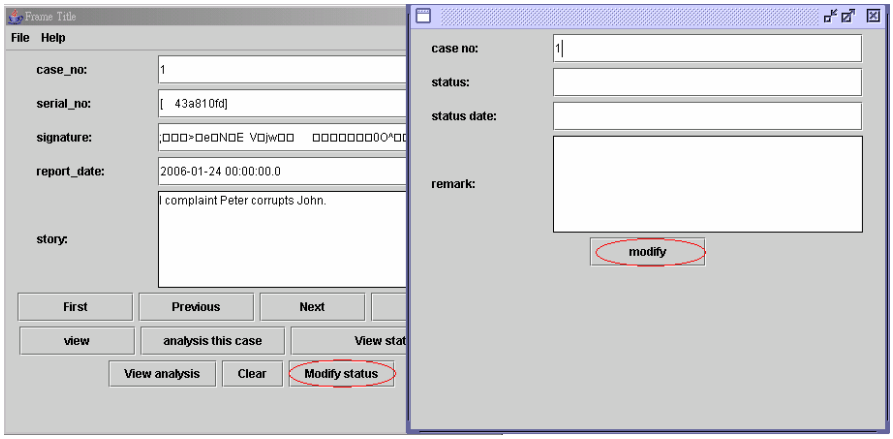
click button “analysis this case” to prompt the analysis screen, C.C.A.C. fill up the analysis information of a specified case and click button “save” to save it into database.



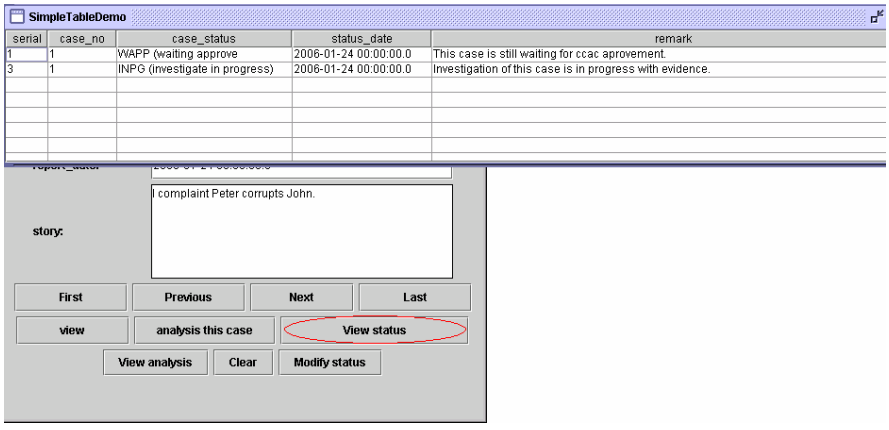
click the “view analysis” button to view back the analysis info of a case.



click the “modify status” button to prompt the status screen, C.C.A.C. fill the updated status of that case and save it.



click the “view status” button to view the status history of a specified case.

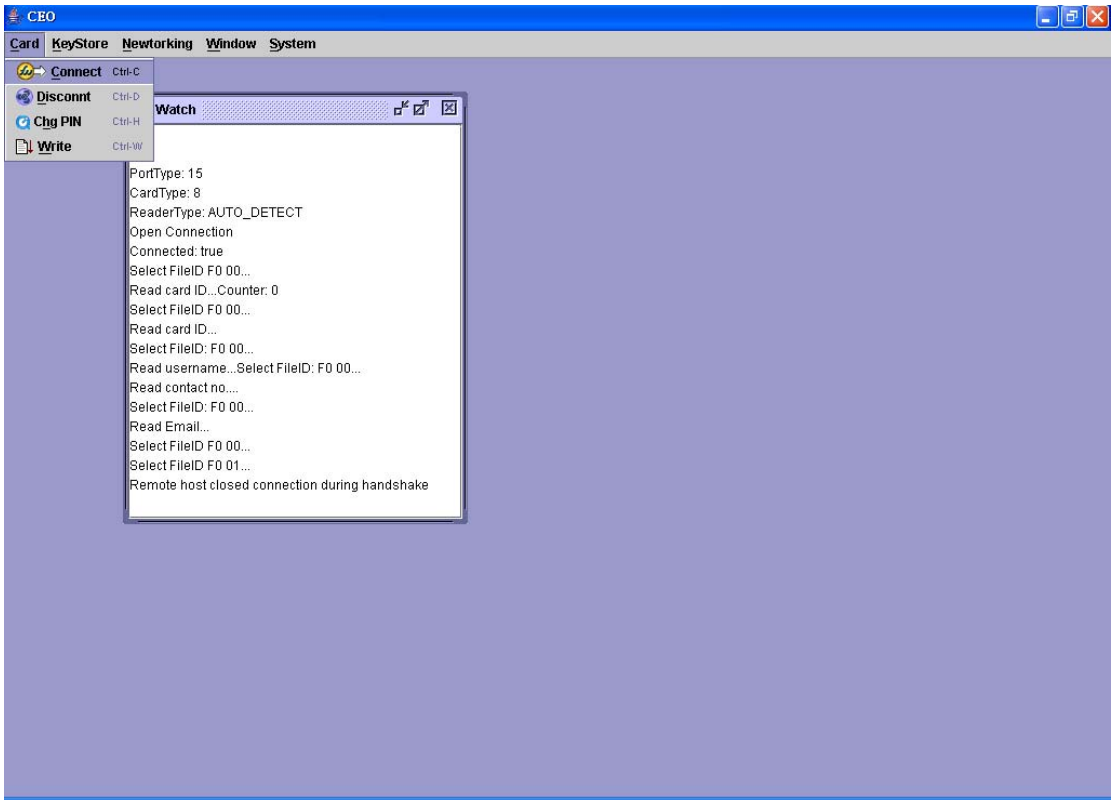


Client

We use the application to do the following tasks if you have access right:

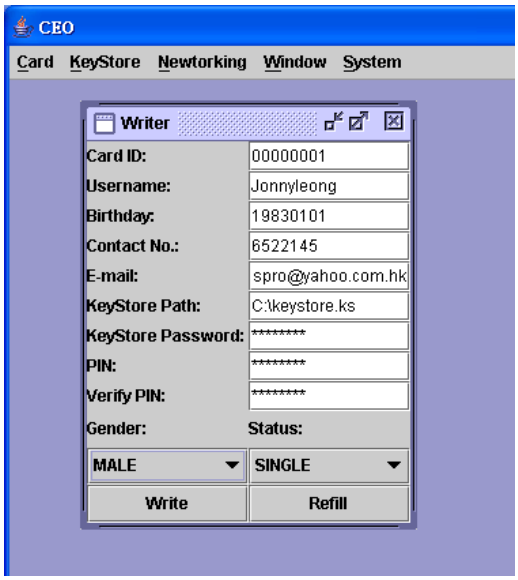
1. Extract certificate.
2. Read user information.
3. Write user information.
4. Change PIN code.
5. Extract KeyStore
6. Report the cases

After our program is run, your will see the following screen



The “Watch” internal frame will show you system exception and what the program is doing, e.g. to show you connected to Smart Card Reader successfully, read the card, now is reading/writing card ID, contact number and KeyStore and so on.

Write User information



Clicked menu item Card -> Connect to communicate Smart Card Reader with Smart Card, after connection success, clicked menu item Card -> Write, you will see the screen, just write down your information and then click “Write” button to save your information to Smart Card.

NOTES:

The PIN code must be at less 8 characters or numbers.

We set the reminder that if fail to provide PIN code 3 times, the program will disconnect to the Reader and fail to provide PIN code 3 times again, the program will disconnect and the card is locked.

Change PIN code

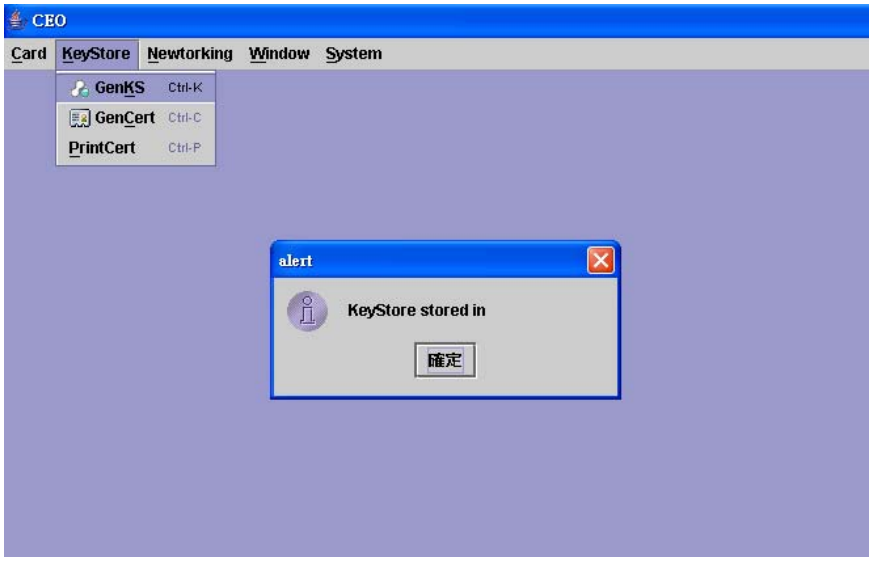
We provide change PIN function to change your Personal Identification Number (PIN).

NOTES:

The PIN code must be at less 8 characters or numbers.

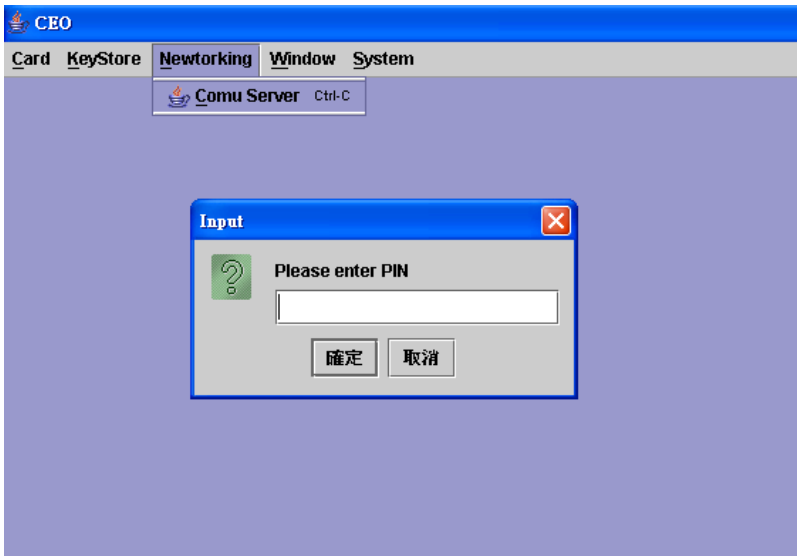
We set the reminder that if fail to provide PIN code 3 times, the program will disconnect to the Reader and fail to provide PIN code 3 times again, the program will disconnect and the card is locked.

Extract KeyStore



After you read all information from Smart Card, you can export our KeyStore, click KeyStore -> GenKS to save your KeyStore, the program will prompt you what path you save KeyStore.

Communication with Server and report the cases



Clicked Networking -> Com Server, this will communicate to Server for Certificate exchange and send user information read from Smart Card and do three way handshakes in SSL environment. Because it refers all information from Smart Card and communicate with Server, the program will prompt you enter PIN, KeyStore password, KeyStore alias and server IP.

In our example, we have the following information;

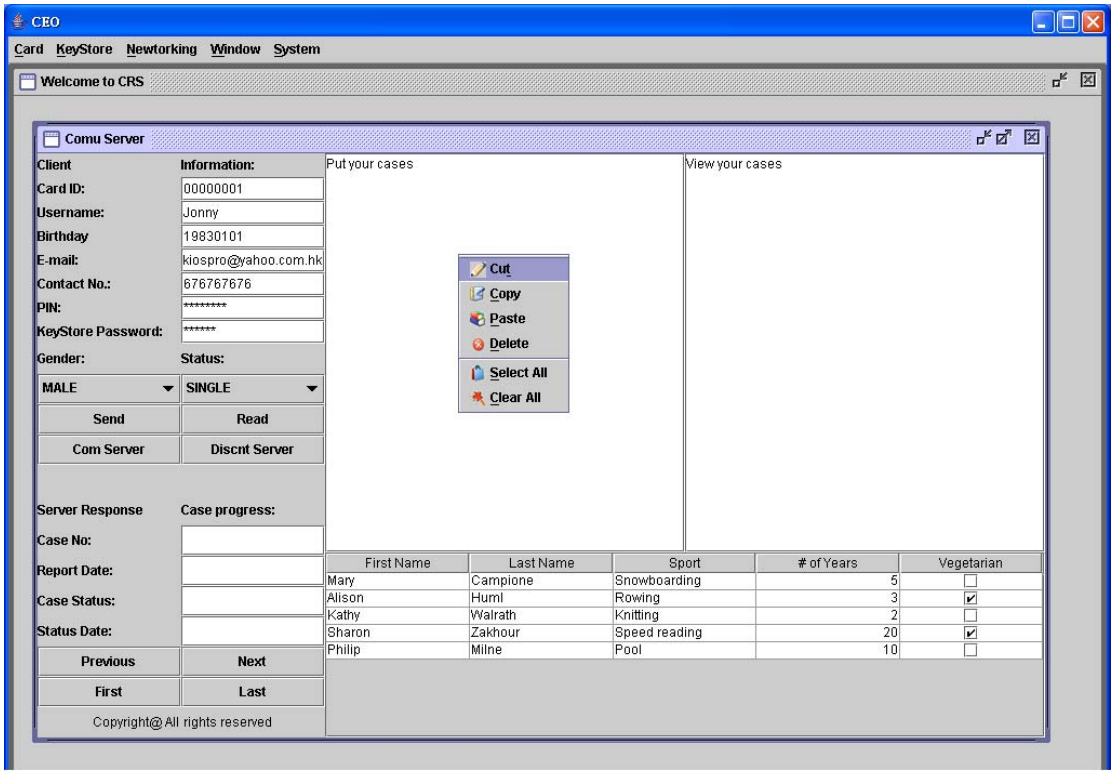
PIN: 11111111

KeyStore password: 123456

KeyStore alias: client

Server IP: localhost

Report cases

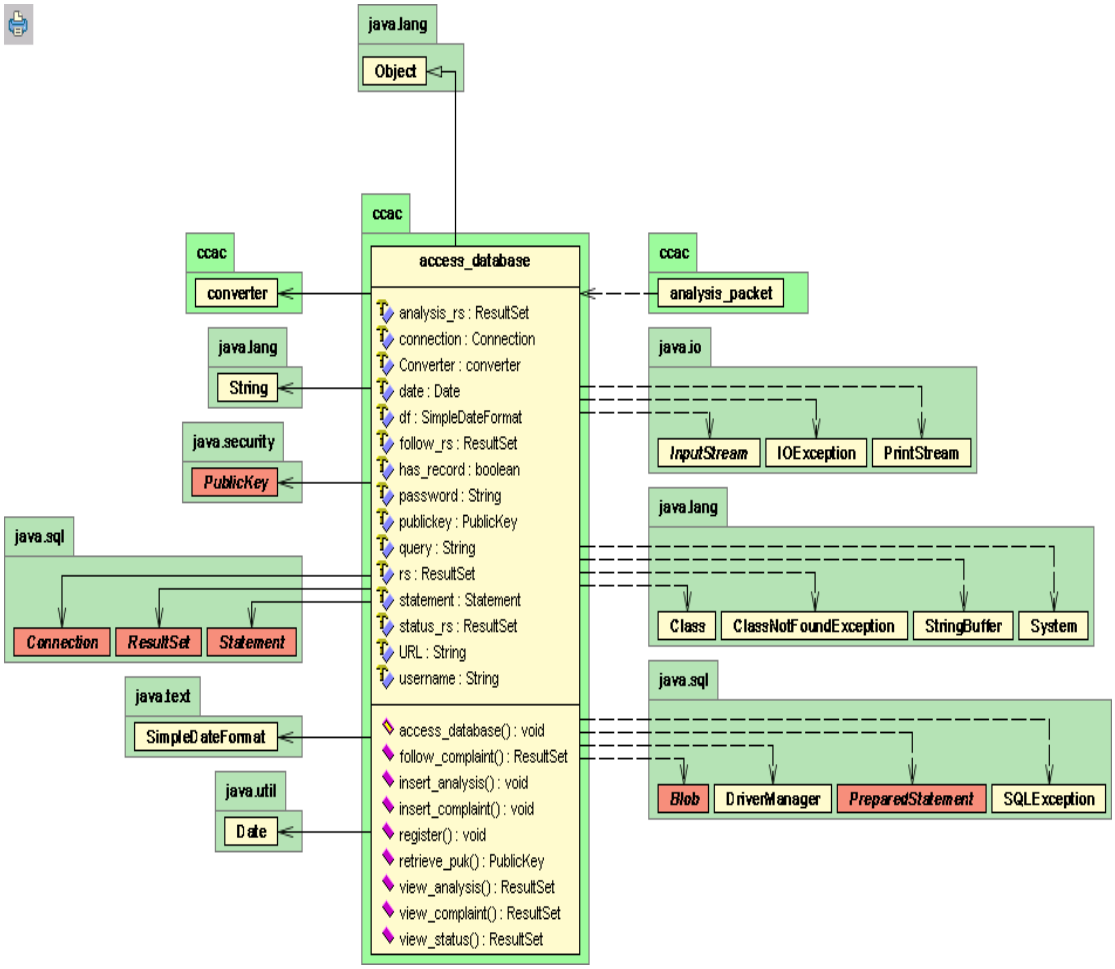


After connection to server, you can see the above screen to show user information, case reported by user and case responded by server.

✸Class structure and illustration

Server

access_database.java



Constructor Detail

Method Detail

register

public void register(String serial, String name, String gender,
String birth, String contact, String email, PublicKey puk)

To check whether residents have registered and sign them up if they are first time use the application.

---

insert_analysis

public void insert_analysis (String caseno, String pleader, String suspect,
String acts, String evidence, String datefrom,

String dateto)

The C.C.A.C. officers analysis the reported case received from the residents and insert the analysis information into the database.

---

retrieve_puk

public PublicKey retrieve_puk (String serial_no)

To retrieve residents' public keys from the database.

---

register

public void register(String serial, String name, String gender, String birth,
String contact, String email, PublicKey puk)

To check whether residents have registered and sign them up if they are first time use the application.

---

insert_complaint

public void insert_complaint (String serialno, byte[] sign, String complaint)

To insert case content reported by residents into database.

---

follow_complaint

public ResultSet follow_complaint (String sql)

To query specified reported case and return its content and status to the residents.

---

view_complaint

public ResultSet view_complaint (String sql)

To query reported cases and return to C.C.A.C. officers. Then C.C.A.C officers can make further analysis and followings.

---

view_status

public ResultSet view_status(String sql)

To query status information of a specified case from the database.

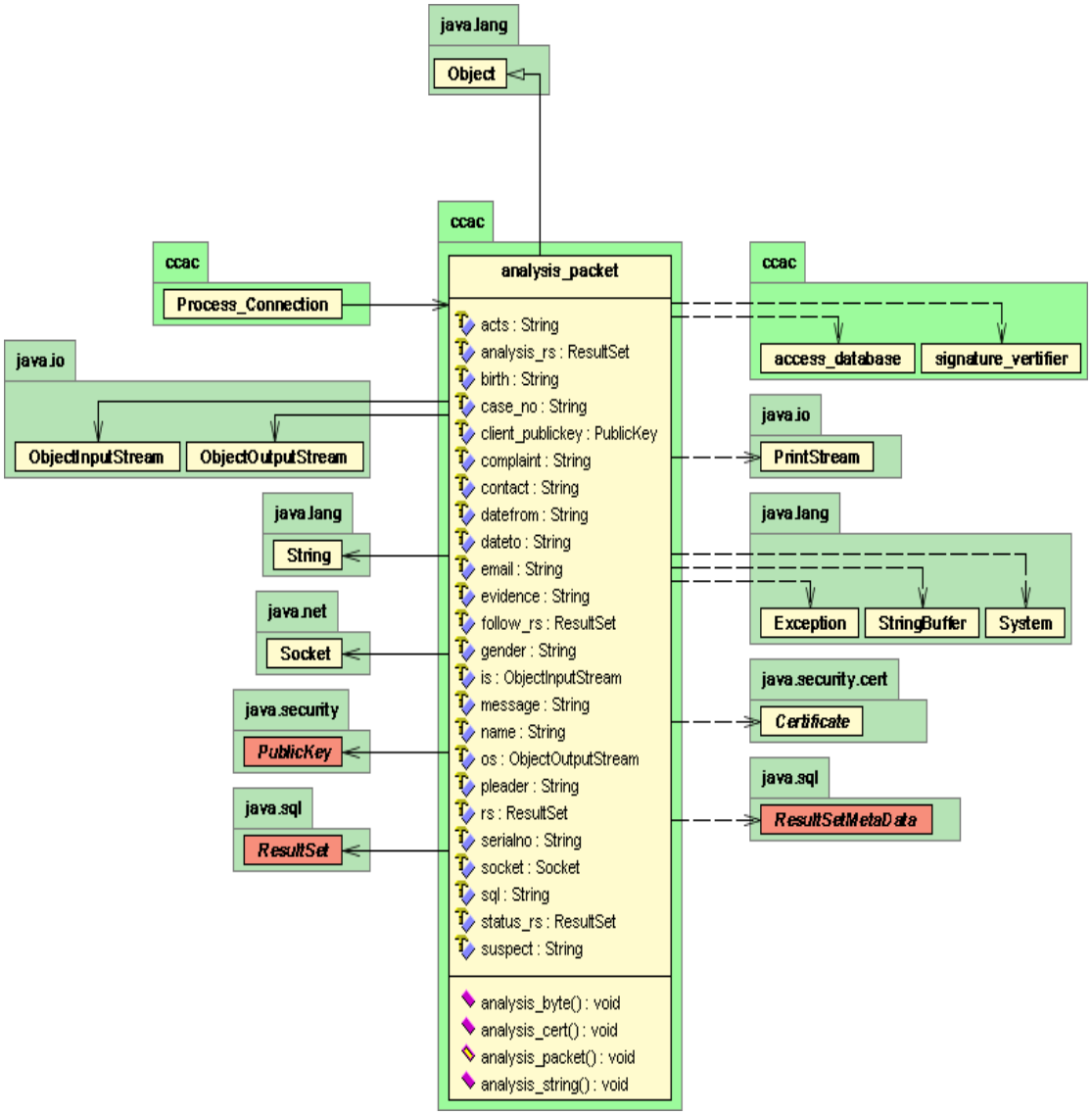
---

view_analysis

public ResultSet view_analysis(String sql)

To query analysis information provided by C.C.A.C. officers and return to only C.C.A.C. officers.

Analysis_packet.java



Constructor Detail

Method Detail

analysis_string

public void analysis_string(String string, Socket clientsocket,
ObjectInputStream in, ObjectOutputStream out)

To analysis the string header in order to determine what kind of request the client is asking for or providing to.

Details are shown as below:

a) header: "vi_ana"

To notify the server it is a request for analysis information of a case from the C.C.A.C. officers.

b) header: "follow"

To notify the server it is a request for case content and status from the resident who reported that case.

c) header: "status"

To notify the server it is a request for current status information of a case and let C.C.A.C. officers change its status.

d) header: "_firs_"

To notify the server it is a request for the first record set of the query result.

e) header: "_last_"

To notify the server it is a request for the last record set of the query result.

f) header: "_prev_"

To notify the server it is a request for the previous record set of the query result.

g) header: "_next_"

To notify the server it is a request for the next record set of the query result.

h) header: “_view_”

To notify the server it is a request for case content by the C.C.A.C. officers for further analysis and followings.

i) header "compla"

To notify the server what it has received is the content of a case reported by a resident.

j) header "serial"

To notify the server what it has received is the unique serial number of resident’s certificate.

k) header "_name_"

To notify the server what it has received is the name of the resident who reported the case.

l) header "gender"

To notify the server what it has received is the gender of the resident who reported the case.

m) header "_birth"

To notify the server what it has received is the birth of the resident who reported the case.

header "contac"

To notify the server what it has received is the contact number of the resident who reported the case.

n) header "_email"

To notify the server what it has received is the email address of the resident who reported the case.

o) header "_case_"

To notify the server what it has received is the case number of which case the C.C.A.C. has analyzed.

p) header "pleade"

To notify the server what it has received is the pleader name of which case the C.C.A.C. has analyzed.

q) header "suspec"

To notify the server what it has received is the suspect name of which case the C.C.A.C. has analyzed.

header: "_acts_"

To notify the server what it has received is the acts information of which case the C.C.A.C. has analyzed.

r) header "eviden"

To notify the server what it has received is the evidence of which case the C.C.A.C. has found during the analysis.

s) header “_dfrom”

To notify the server what it has received is the date of case begins that the C.C.A.C. has found during the analysis.

t) header "_dato_"

To notify the server what it has received is the date of case ends that the C.C.A.C. has found during the analysis.

u) header "analys"

To notify the server it is a request to insert the analysis information provided by C.C.A.C. officers into the database.

---

analysis_cert

public void analysis_cert(Certificate cert)

To notify the server it is a request of registering a new user who is the first time using the application.

---

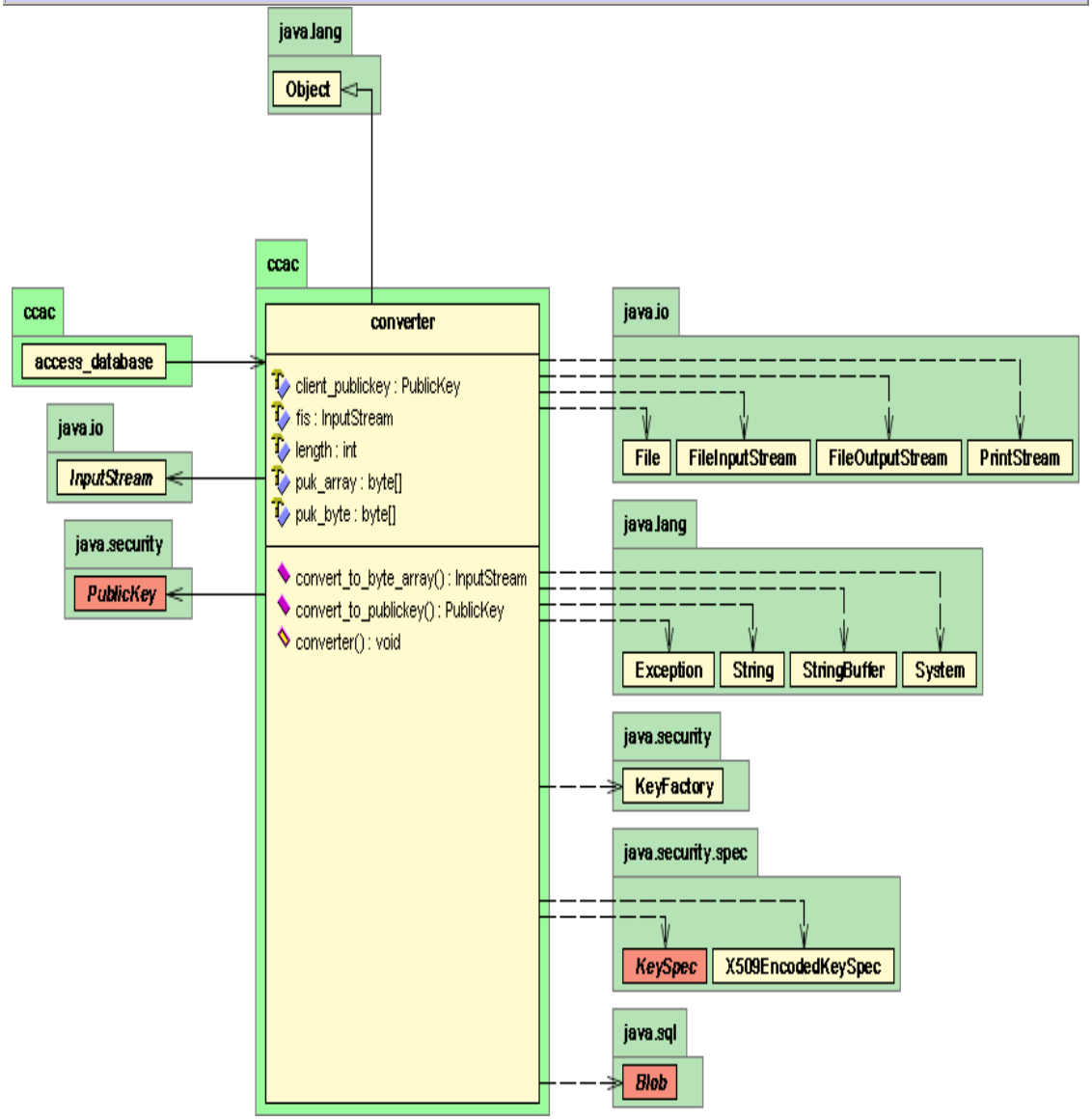
analysis_byte

public void analysis_byte (byte[] sig)

To notify the server it received the report case with it digital signature. The server verifies the signature and stores the report cases into the database.

---

converter.java



Constructor Detail

Method Detail

convert_to_byte_array

public InputStream convert_to_byte_array(PublicKey puk)

To convert resident's public key into file input streaming (byte array)

---

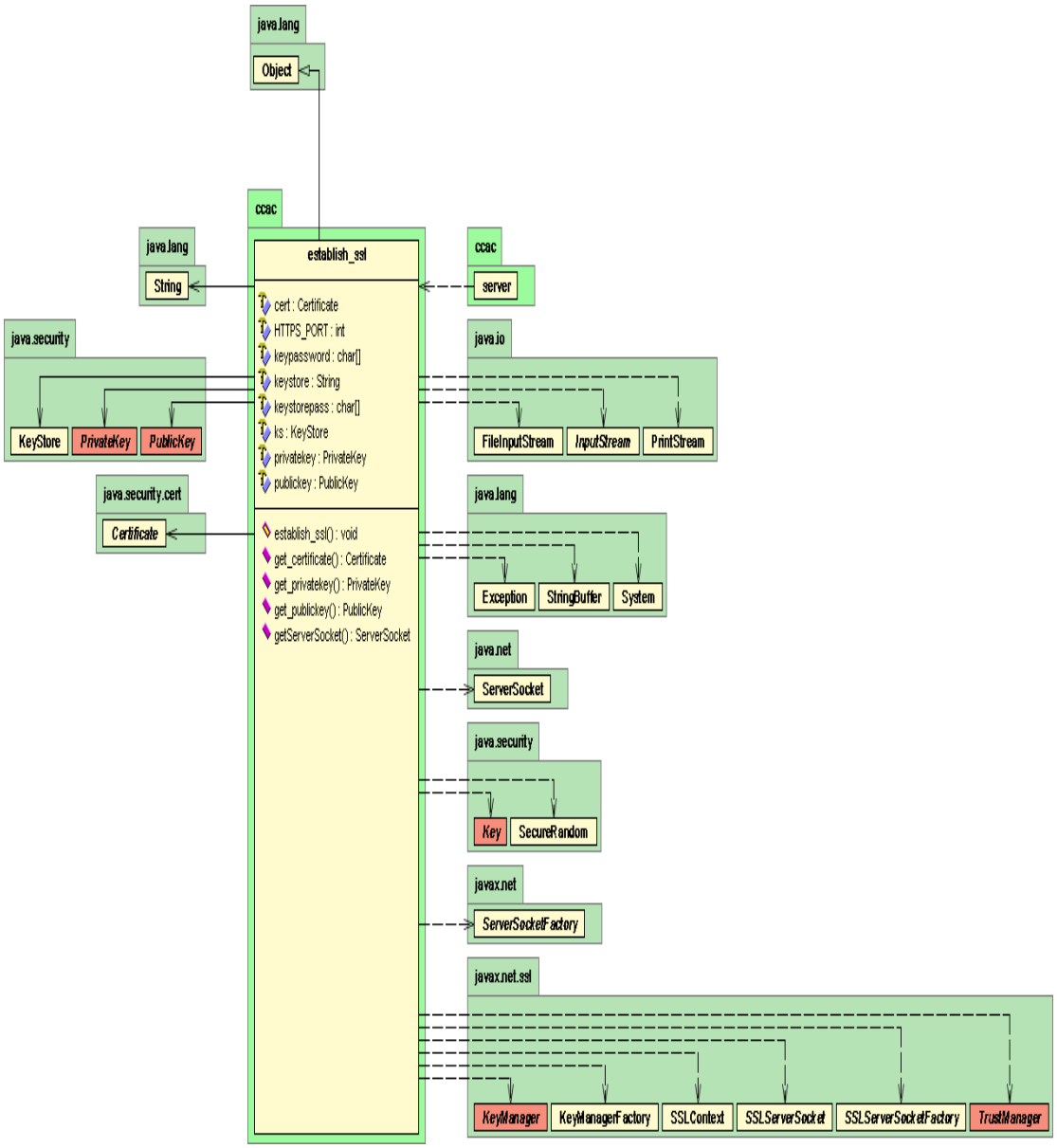
convert_to_publickey

public PublicKey convert_to_publickey(Blob blob)

To convert the Blob object back to public key object.

---

establish_ssl.java



Constructor Detail

Method Detail

getServerSocket

public ServerSocket getServerSocket ()

To create socket side SSL-enable socket with its key store.

---

get_certificate

public Certificate get_certificate()

- To prepare server’s certificate

get_privatekey

public PrivateKey get_privatekey()

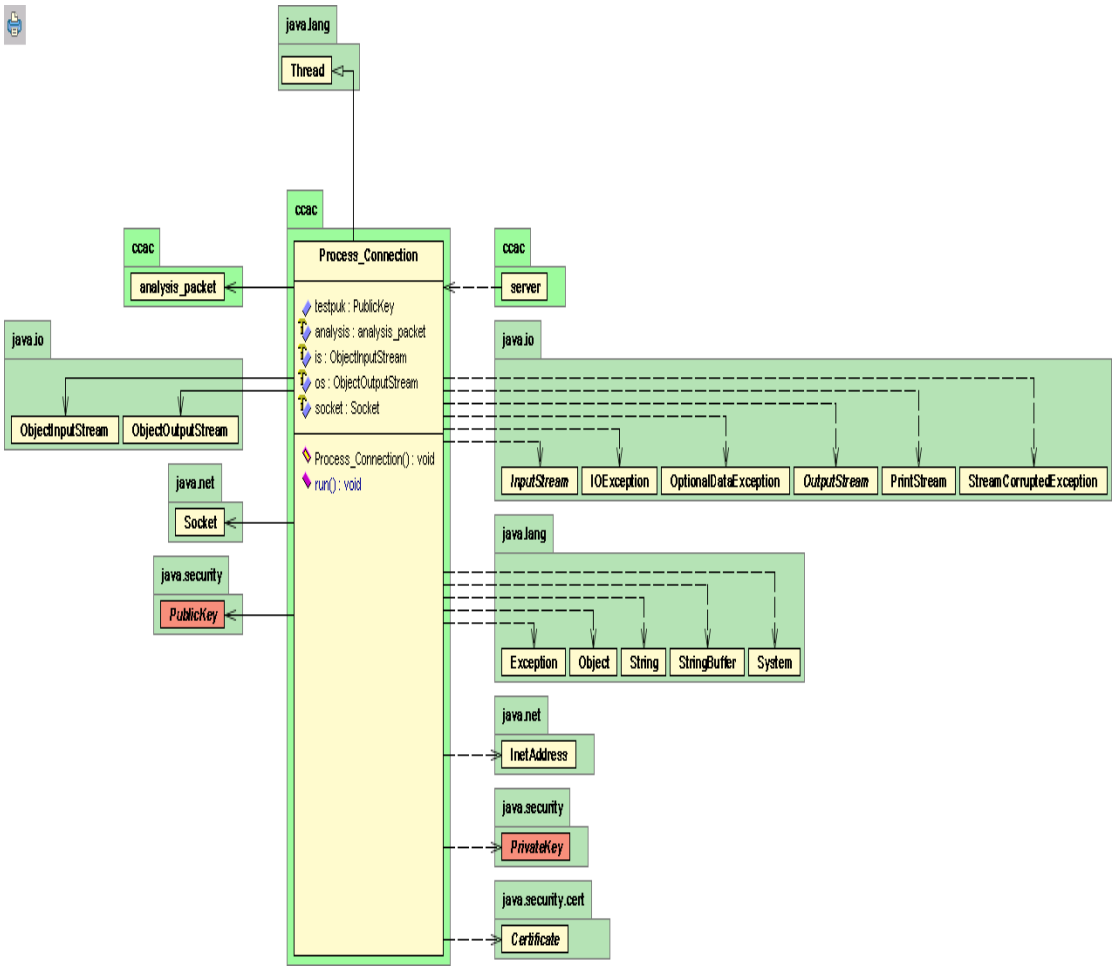
To retrieve server's private key from its key store.

get_publickey

public PublicKey get_publickey()

To retrieve server's public key from its key store.

Process_Connection.java



Constructor Detail

Process_Connection

public Process_Connection(Socket clientsocket, PublicKey server_public_key, PrivateKey server_private_key)

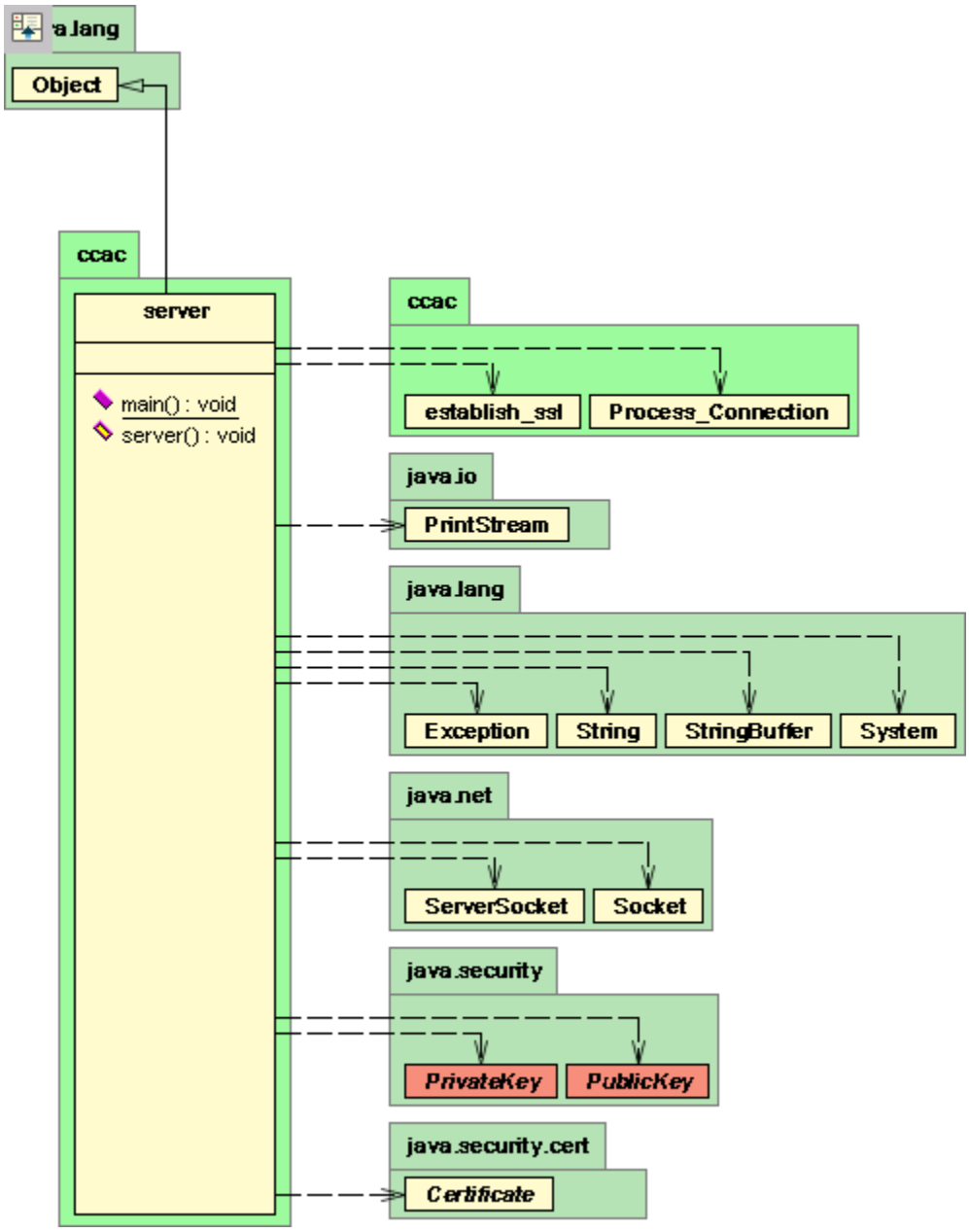
To create input stream and output stream between the server and a individual client.

Method Detail

public void run()

To receive and process requests from a client.

server.java



Constructor Detail

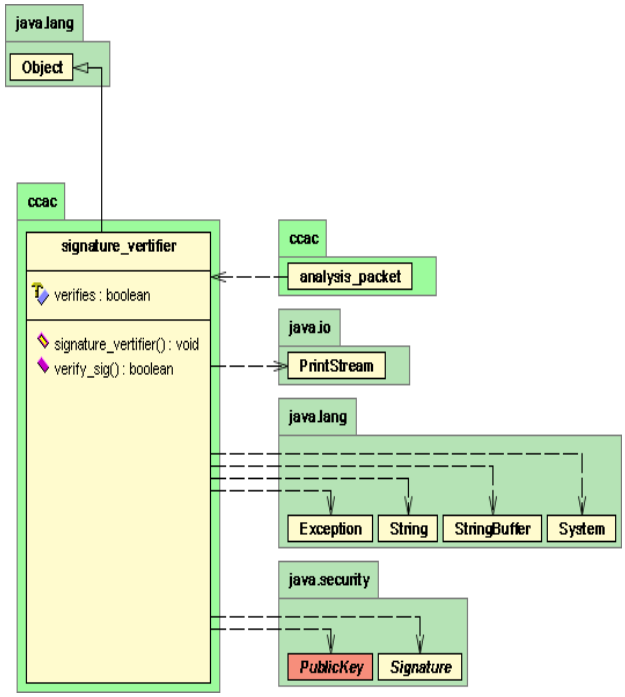
server()

public server()

Waiting for a connection request from a client and unitized the SSL Handshaking with the server public certificate to complete the server and the client mutual

authentication.

Signature_verifier.java



Constructor Detail

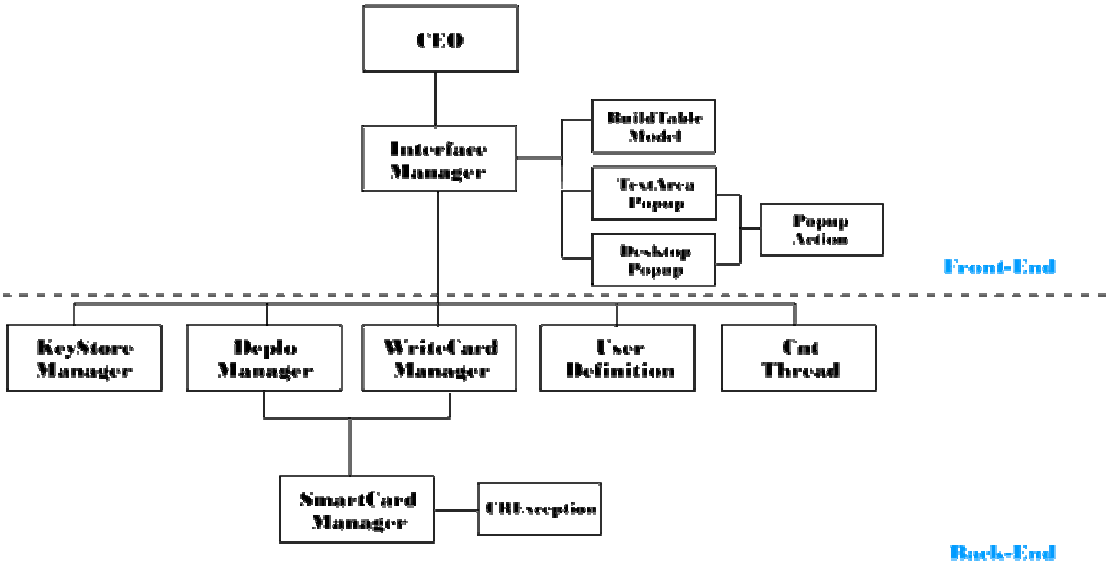
Method Detail

verify_sig

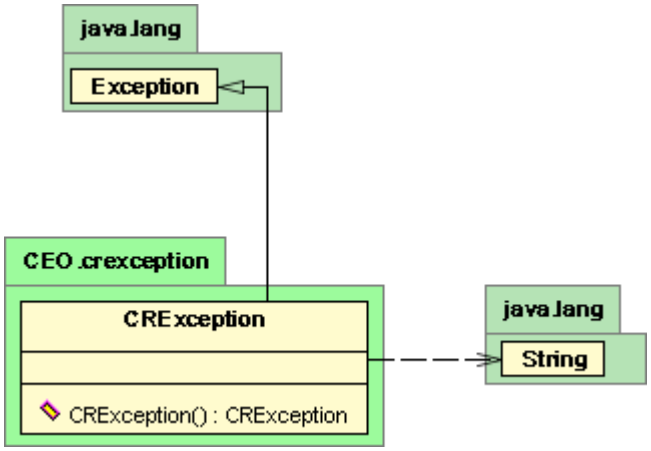
public boolean verify_sig (byte[] sig, PublicKey puk, String complaint)

To verify the digital signature with the resident's public key in order to ensure the dentify of the case reporter. It makes the C.C.A.C. officers knew who reported the case.

Client



CRException.java



Constructor Detail

CRException

public CRException(String msg)

Construct an exception of card and reader

Parameters:

msg – the detail message

SmartCardManager.java

Constructor Detail

SmartCardManager

public SmartCardManager(int port, int cardType)

Construct a class that use JNI to communicate with Card Reader, called functions in API in C language.

Parameters:

port – to set the port type.

AC_COM1	0
AC_COM2	1
AC_COM3	2
AC_COM4	3
AC_COM5	4
AC_COM6	5
AC_COM7	6
AC_COM8	7
AC_KEYB	16
AC_PCSC	20
AC_USB	15
AC_USB1	15
AC_USB2	30
AC_USB3	31
AC_USB4	32

cardType – to set card type

AC_AM104	1
AC_AM221	2
AC_SLE4404	3
AC_GPM896	4
AC_AT101	5
AC_AT102	6
AC_AM8KP	7
AC_AM8KS	8
AC_AM2KP	9
AC_AM2KS	10
AC_IIC	11
AC_XIIC	12
AC_AT1604	13
AC_T0	14

AC_T1	15
AC_SCModule	16
AC_AM256	17
AC_AM4KP	18
AC_X76F041	19
AC_X24645	20
AC_ST1335	21
AC_ST1333	22
AC_X76F128	23
AC_X76F640	24
AC_X76F100	25
AC_AT45D041	26
AC_AT45DB041	27

Method Detail

changePIN

public boolean changePIN(String oldPin, String newPin)

to change Personal Identification Number (PIN) in Smart Card.

Parameters:

oldPin – the original PIN code.

newPin – the new PIN code.

---

initCardStructure

public void initCardStructure()

Construct the user file structure of the card.

---

initUserDataFile

public void initUserDataFile(byte rid, int length)

Initializes user data file in the card

Parameters:

rid – record ID

length – the record length

---

isCardInserted

public int isCardInserted()

to detected the card is inserted or not

Returns:

return true if card is inserted.

---

readXXX

public String / byte[] readXXX(String XXX)

XXX represents different user informations,

e.g. readCardId(), readUsername(), readEmail(), readContactNo(),

readOthers(), readKeyStore()

Returns:

User information and KeyStore in byte array

---

writeXXX

public void writeXXX(String XXX)

XXX represents different user informations,

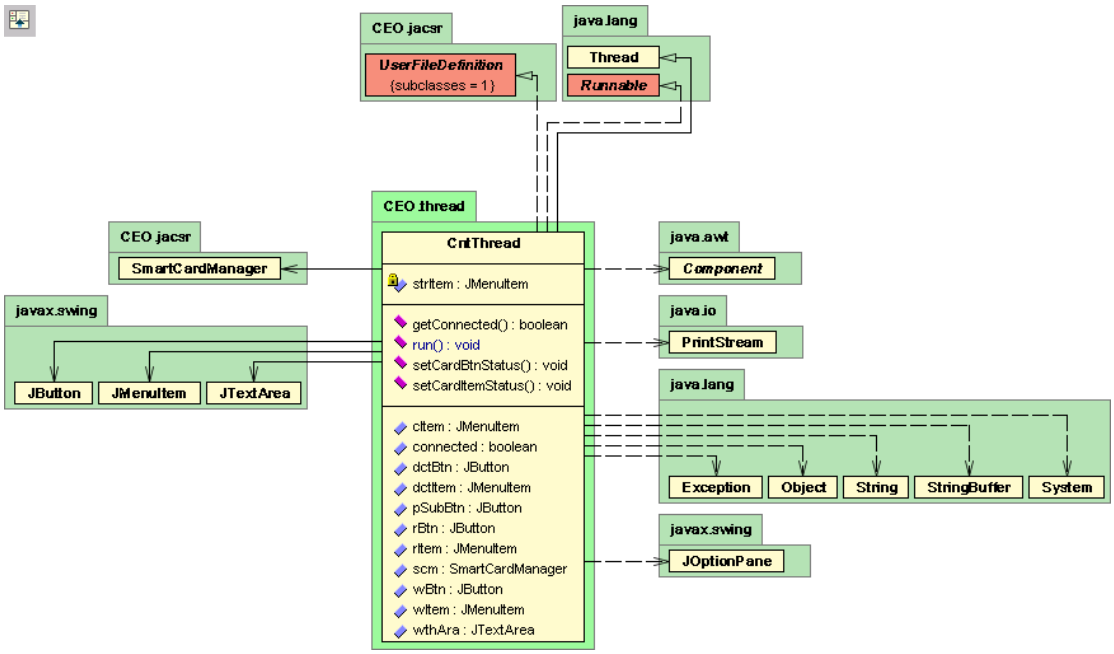
e.g. writeCardId(String cardId), writeUsername(String username),

writeEmail(String email), writeContactNo(String contactNo),

writeOthers(int gender, int age, int status, int level, int ksLen),

writeKeyStore(byte[] keystoreInByte)

CntThread.java



Constructor Detail

Extended Thread

Method Detail

run

public void run()

This method detects the card is taken out.

---

getConnected

public boolean getConnected()

Detected the card is inserted.

Returns:

If the card inserted return true.

---

setConnected

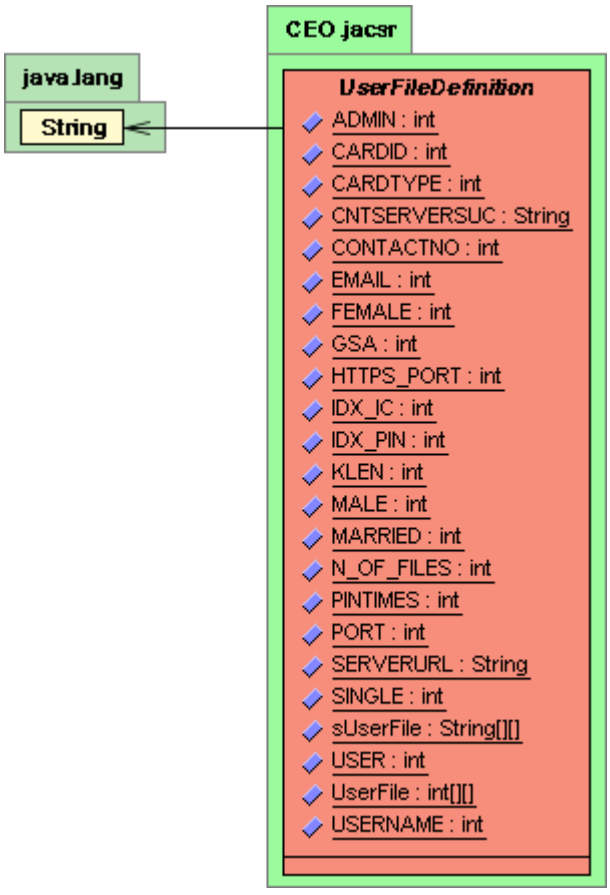
public setConnected(boolean connected)

Detected the card is inserted.

Parameters:

set the card is inserted.

UserDataDefinition.java



public interface UserFileDefinition

To mark down static final parameters.

WriteCardManager.java

Constructor Detail

WriteCardManager

public WriteCardManager()

Method Detail

writeCardInfo

public void writeCardInfo(String pin, String ksPwd)

Write user information, card information and KeyStore to the Smart Card.

Parameters:

pin – PIN code.

ksPwd – KeyStore password.

---

isBlankCard

public boolean isBlankCard()

To recognize the card is blank or not.

Returns:

Return true if the card is blank.

DeploManager.java

Constructor Detail

DeploManager

public DeploManager()

It is a diplomat to negotiate with server and read information from Smart Card.

Method Detail

checkPin

public boolean checkPin(String pin)

Check PIN is valid or not, if invalid PIN entered three times, disconnected to card reader, if invalid PIN entered three times again, Smart Card is locked.

Parameters:

pin – the PIN code in the Smart Card.

Returns:

It is a correct PIN if returns true

---

connectToServer

public boolean connectToServer(String serverip,

KeyStore ks, String alias, String ksPwd)

To establish connection to server

Parameters:

serverip - Server IP.

ks – KeyStore.

alias – the alias of the KeyStore.

ksPwd – the password of the KeyStore.

Returns:

Return true if connection successful.

---

readCardInfo

public boolean readCardInfo(String pin, String ksPwd)

to reads information from Smart Card.

Parameters:

pin - PIN code.

ksPwd – KeyStore password.

Returns:

Return true if reads card successfully.

---

sendComplaint

public void sendComplaint()

To send complaint to server.

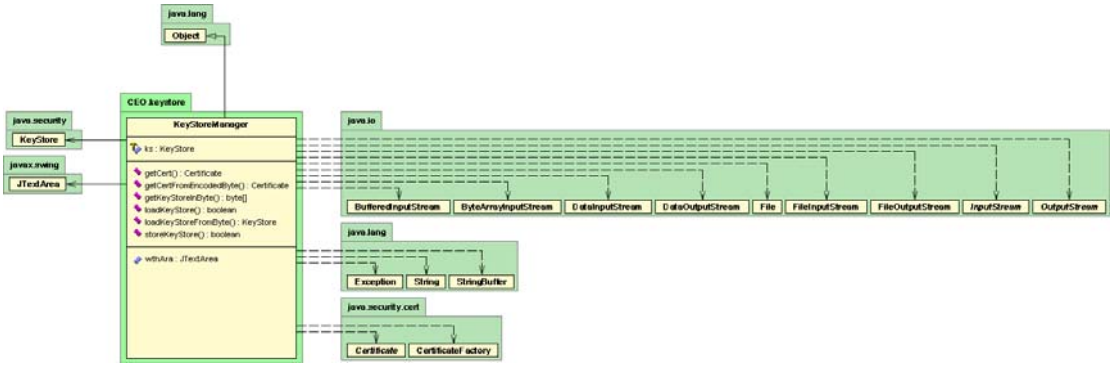
---

viewComplaint

public void viewComplaint()

To view complaint responses by server.

KeyStoreManager.java



Constructor Detail

To manage KeyStore

Method Detail

getKeyStoreInByte

public byte[] getKeyStoreInByte(String path, String ksPwd)

To convert KeyStore to byte array

Parameters:

path – KeyStore path.

ksPwd – KeyStore password.

Returns:

Return byte array of the KeyStore.

---

loadKeyStoreFromByte

public KeyStore loadKeyStoreFromByte(byte[] keystoreInByte, String kspwd)

To convert KeyStore from byte array to KeyStore type

Parameters:

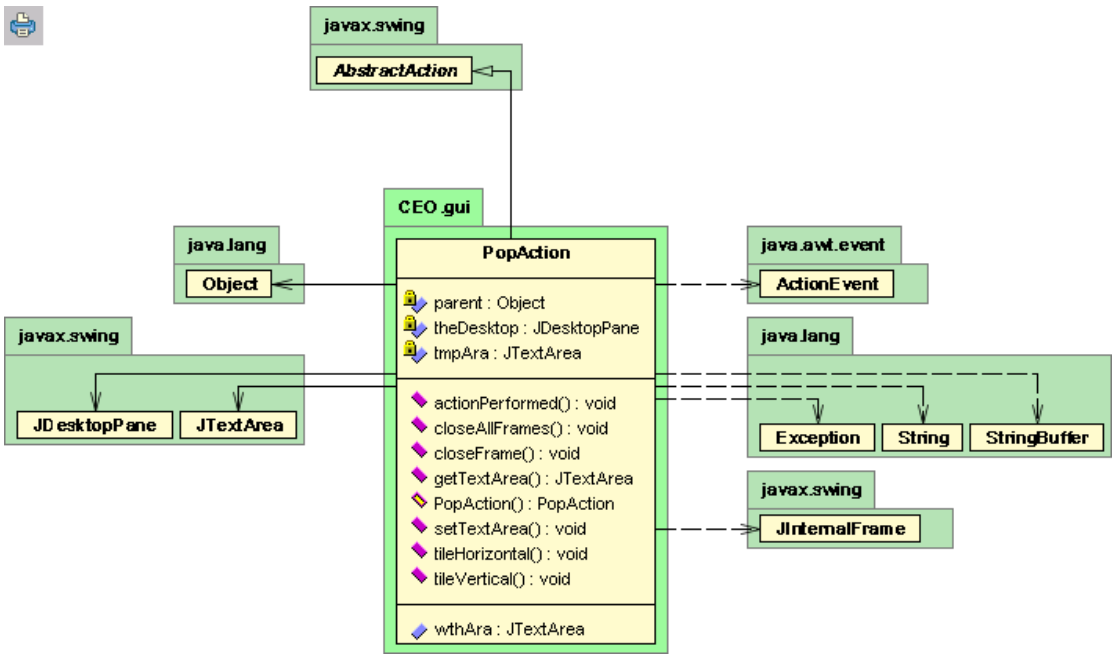
keystoreInByte – the byte array of KeyStore array

ksPwd - KeyStore password.

Returns:

Return KeyStore converted from byte array.

PopAction.java



Constructor Detail

PopAction

public PopAction(String name, Object parent)

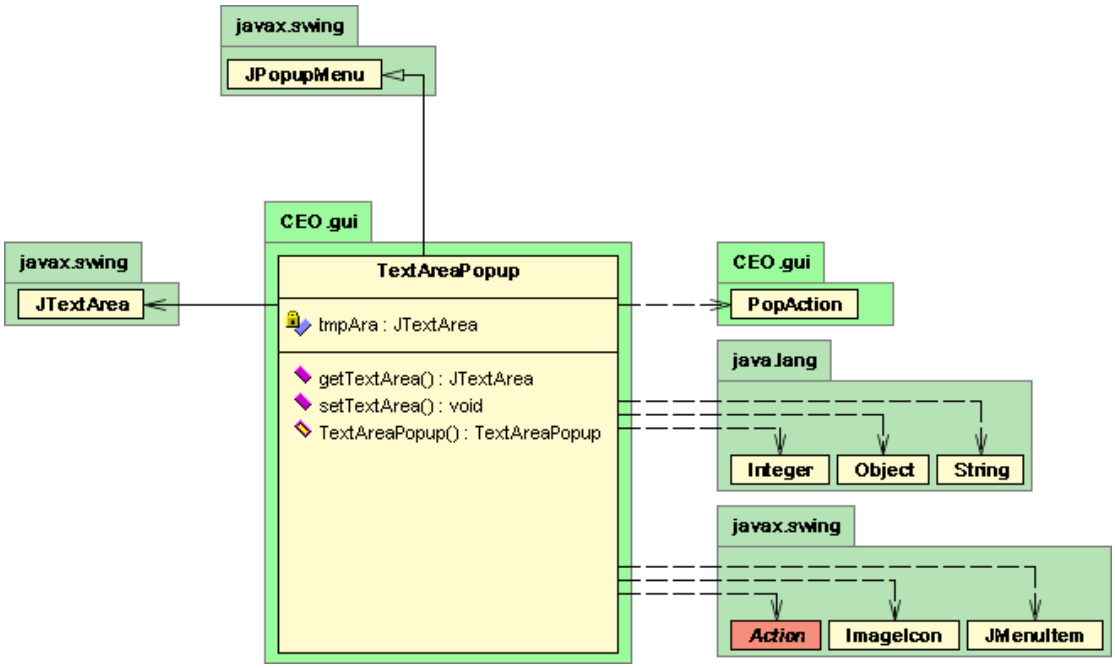
To construct popup menu.

Parameters:

name - The name of popup action.

parent – swing components.

TextAreaPopup.java



Constructor Detail

TextAreaPopup

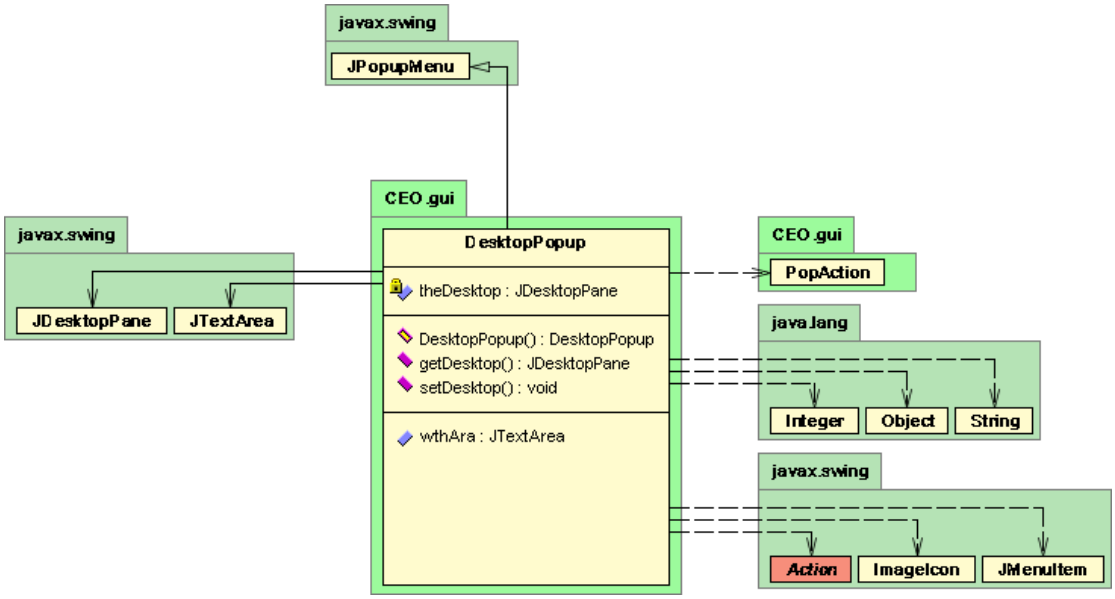
public TextAreaPopup(JTextArea tmpAra)

To construct popup menu for JTextArea.

Parameters:

tmpAra – JTextArea that attached popup menu.

DesktopPopup.java



Constructor Detail

DesktopPopup

public DesktopPopup(JDesktop desktop)

To construct popup menu for JDesktopPane..

Parameters:

desktop – JDesktopPane that attached popup menu.

BuildTableModel.java

Constructor Detail

BuildTableModel

public BuildTableModel(Object[][] data, String[] columnNames)

To build JTable to display cases composed by citizen.

Parameters:

data – cases information.

columnNames – column names.

InterfaceManager.java

Constructor Detail

InterfaceManager

public InterfaceManager()

To create GUI to user.

Method Detail

frameActivate

public void frameActivate(JInternalFrame frame, JMenuItem curItem)

To open JInternalFrame.

Parameters:

frame – the internal frame.

curItem – the menu item to control that frame.

---

closeFrame

public void closeFrame()

To close an activated frame.

---

closeAllFrames

public void closeAllFrames()

To close all frames are displayed.

---

tileVertical

public void tileVertical()

To display the frames with tile vertical

---

tileHorizontal

public void tileHorizontal()

To display the frames with tile horizontal